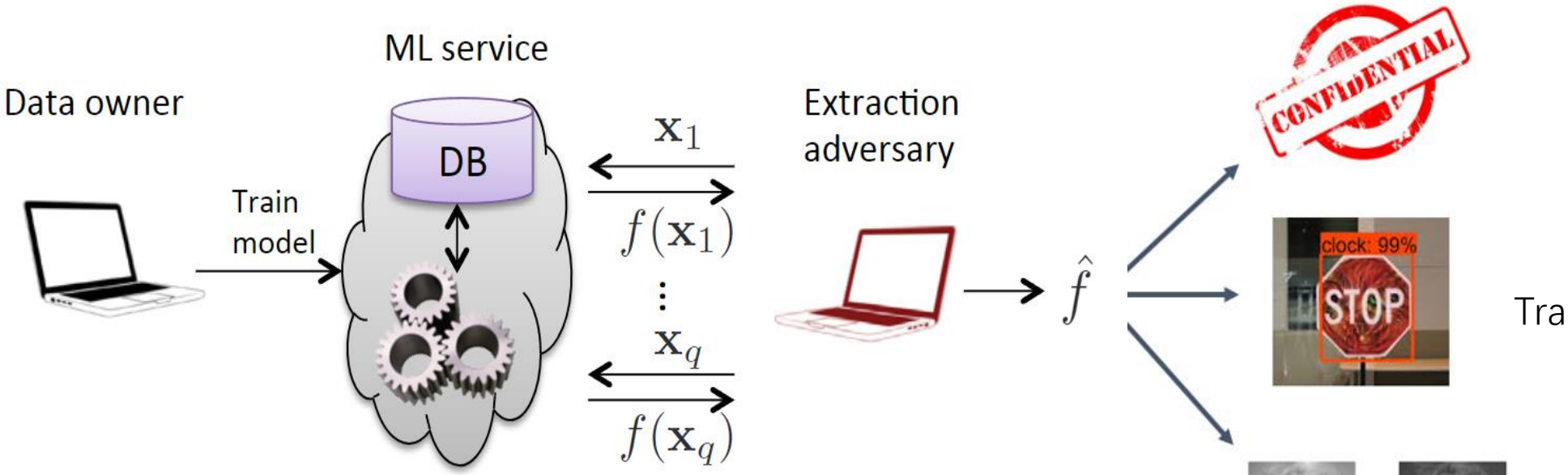
Protecting Object Detection Models From Model Extraction Attack via Feature Space Coverage

Zeyu Li, Yuwen Pu, Xuhong Zhang, Yu Li, Jinbao Li, Shouling Ji

IJCAI 2024

Background

- **Model extraction attack**



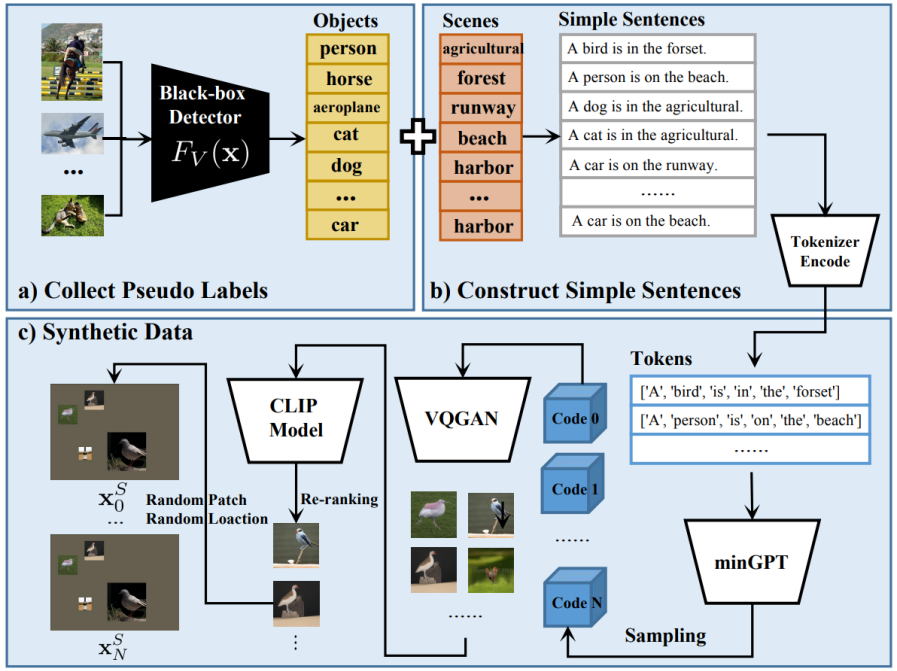
confidentiality

Transfer Adversarial Attack

Model Inversion

Background

- **Model extraction attack and defense for various types of models**
  - Classification models were the primary focus of early research.
  - GNN： SLGNN (USENIX 2021), MSAIGNN (USENIX2022)
  - Generation models， encoders： SMLM (ACSAC 2021), Cont-Steal (CVPR 2023), StolenEncoder (CCS 2022)



Process of data generation in Imitated Detector

- Only Imitated Detector(MM 2022) focused on extracting against object detection models.
- There is still no research focused on model extraction defense for widely used object detection models.
- To bridge this research gap, we explore defense countermeasures explicitly for object detection models and propose a detection method called OSD (Object detector Steal Detection).

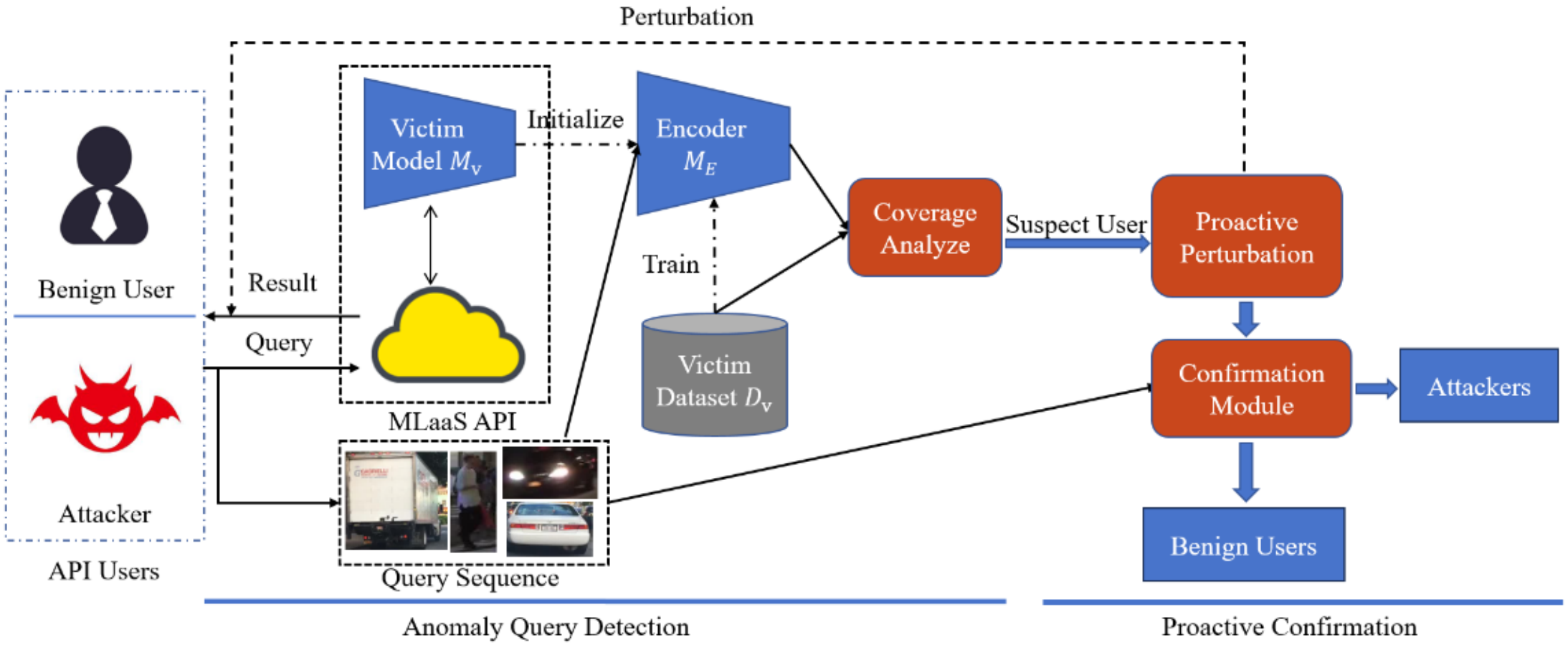
OSD：Background

- **Background**
  - Model extraction defenses primarily consist of proactive defense methods involving proactive perturbations [Mazeika et al., 2022] and passive defense methods such as model watermark and model fingerprint. However, both methods have their limitations.

- **Problem Description**
  - OSD is primarily used in cloud computing platforms. Model owners deploy trained neural network models onto the platform and provide API services to platform users. Model owners have complete access to query samples and the target model.
  - Model owners aim to detect and block attackers among all users by monitoring the query traffic of all users, based on minimal query history. Once an attacker is identified, the model owner can block their access privileges to prevent theft attacks.

- **Requirment:** 1. Accuracy requirement 2. Efficiency requirement
- **Research question:** 1. Defense accuracy 2. Defense efficiency 3. Deployment cost 4. Superority

OSD：Intuition



- **Anomaly query detection**：identification of suspicious users via feature space coverage analysis.
- **Proactive confirmation**：perturb the returned results for suspicious users and determines if they are attackers based on subsequent queries.

OSD：Anomaly Query Detection

In model extraction attacks against object detection models, attackers aim to steal the model by ensuring that their samples contain as many distribution-specific features as possible.

- **Feature space coverage analysis**

  Train a contrastive encoder M_E through contrastive learning, which aims to generate embedded feature vectors for all samples in the target dataset D_v. Apply a similarity threshold θ_A to select anchor points A that are uniformly distributed in the feature space. Calculate the feature space coverage C.

$$C_i = \begin{cases} 1, & max(sim(A_i, M_E(Q))) > \theta_c \\ 0, & otherwise \end{cases}$$

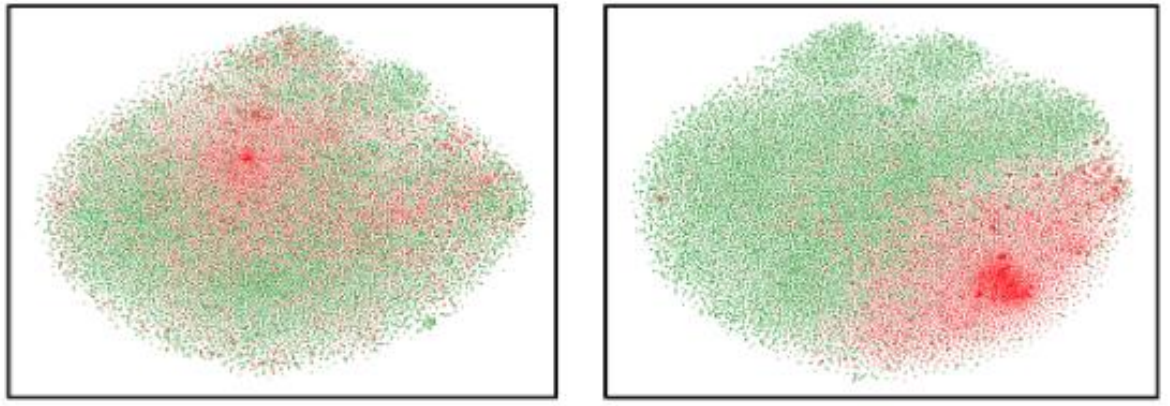
$$Cov(q, A) = \frac{\sum_{i=0}^{c_A} C_i}{c_A}$$

- **Dimension reduction:**

Model	Cosine similarity	t-SNE	IPCA	IPCA*
Raw Sample storage	✗	✗	✗	✗
Embedding storage	✗	✓	✓	✗
Time cost	✗	✓	✗	✗
Representation storage	✗	✗	✗	✓

Table 1: Storage cost comparison between feature similarity calculation methods

The distribution difference between attack samples and normal query samples



(a) attack samples (b) benign samples

OSD：Proactive Confirmation

- **Further reducing the false positive rate based on the active confirmation module.**
  1. Randomly perturb larger-sized objects o in the selected samples.

$$o_{i,x_0} = o_{i,x_0} + rand(-\alpha, \alpha) * (o_{i,x_1} - o_{i,x_0})$$

  2. Analyze the following samples：
     - Perturbing larger-scale objects can lead to a performance degradation of substitute models at that scale.
     - Calculate the proportion P of different-scale objects in the subsequent N_p query samples and compare it with the threshold θ_p.

$$N_p = \eta * N_f \qquad P = (O_l'/O')/(O_l/O)$$

Experiment Setting

- **Model**
  1. Yolov3/5/7
  2. Fast-rcnn
- **Dataset**
  1. Attack dataset : VOC, BDD100K, DOTA
  2. Target dataset D_v: COCO, nuImages, AI-TOD
  3. Normal query dataset: Caltech-101, LVIS, KITTaI, Tusimple, RSOD, VHR-10

- **Other defense method**
  1. HODA
  2. OOD Detector

Experiment

Detection accuracy & false positive rate: Defense detection accuracy using two comparison methods, cosine similarity and IPCA*.

Model	nuImages				COCO				AI-TOD			
	cosine		IPCA*		cosine		IPCA*		cosine		IPCA*	
	ACC	FPR	ACC	FPR	ACC	FPR	ACC	FPR	ACC	FPR	ACC	FPR
YOLOv3	0.893	0.1	0.893	0.05	0.857	0.15	0.929	0.10	0.857	0.00	0.929	0.00
YOLOv5	0.964	0.05	0.964	0.05	0.929	0.10	1.000	0.00	0.964	0.05	0.964	0.00
YOLOv7	0.929	0.05	1.000	0.00	0.893	0.05	0.929	0.05	0.929	0.00	0.893	0.05
Fast-RCNN	0.857	0.15	0.857	0.10	0.893	0.05	0.857	0.05	0.857	0.10	0.929	0.05

Table 2: Detection accuracy evaluation using cosine similarity and IPCA*

- OSD demonstrates high accuracy in detecting extraction against object detection models.
- In the detection of extraction attacks against object detection models, OSD outperforms other comparison methods in terms of performance.

Model		nuImages		COCO	
		ACC	FPR	ACC	FPR
YOLOv3	OOD	0.392	0.65	0.571	0.40
	HODA	0.429	0.70	0.679	0.45
	OSD	**0.893**	**0.05**	**0.929**	**0.10**
YOLOv5	OOD	0.429	0.55	0.464	0.5
	HODA	0.464	0.75	0.643	0.4
	OSD	**0.964**	**0.05**	**1.000**	**0.00**

Table 5: Defense effect comparison with other monitor-based defense methods

Experiment

Defense efficiency of OSD

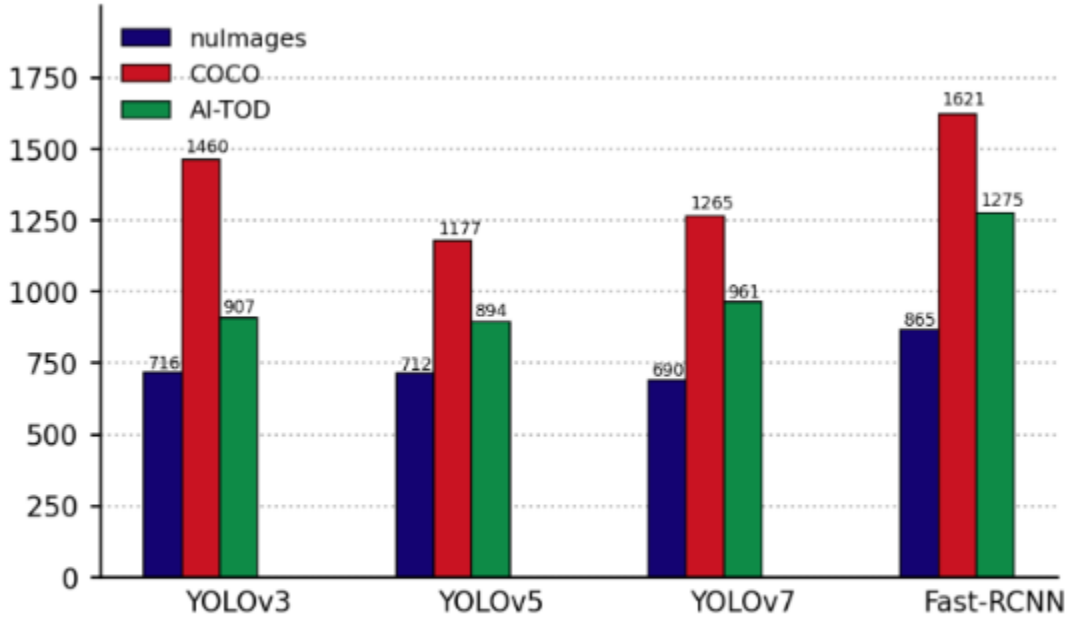


Figure 3: The number of query before detected

Model	nuImages	COCO	AI-TOD
YOLOv3	0.086(11.5%) 0.035	0.055(8.42%) 0.026	0.074(14.3%) 0.031
YOLOv5	0.083(11.5%) 0.025	0.043(6.71%) 0.023	0.061(11.8%) 0.027
YOLOv7	0.092(11.6%) 0.040	0.075(10.8%) 0.039	0.064(12.0%) 0.033
Fast-RCNN	0.076(12.3%) 0.021	0.046(8.30%) 0.017	0.037(7.5%) 0.019

Table 3: The performance of the substitute model when detected. The two rows for each model architecture correspond to metric: mAP50, mAP50:95

- Attackers can be accurately identified with a minimal number of queries, and the substitute model becomes unavailable upon detection.

Experiment

Computation time cost: Comparison of time costs for querying 1000 times between native API and deploying the defense.

Adapt attack： In the defense framework based on feature space coverage, attackers can bypass the detection by adding background images that do not contain the objects or by using query samples that only include common objects.

Model	nuImages			COCO		
	original(s)	defense(s)	δ	original(s)	defense(s)	δ
YOLOv3	159	181	14.6%	136	159	16.9%
YOLOv5	134	157	17.2%	115	141	22.6%
YOLOv7	115	136	18.3%	110	127	15.5%
Fast-RCNN	351	385	9.7%	298	329	10.4%

Table 4: Computation time cost of OSD deployment

Model	nuImages		COCO		AI-TOD	
	ACC	FPR	ACC	FPR	ACC	FPR
YOLOv3	0.929	0.10	0.893	0.15	0.929	0.10
YOLOv5	0.893	0.15	0.857	0.20	0.893	0.15
YOLOv7	0.964	0.05	0.929	0.10	0.929	0.10
Fast-RCNN	0.785	0.250	0.821	0.20	0.857	0.15

Table 6: The ablation study results

- The deployment of OSD incurs minimal time overhead, which aligns with practical requirements.
- Under adaptive attacks, the defense effectiveness of OSD may experience some degradation but still retains strong defense capabilities.

Experiment

Ablation study: Discriminating attackers using only the first stage of the anomaly query detection module.

Model	nuImages		COCO		AI-TOD	
	ACC	FPR	ACC	FPR	ACC	FPR
YOLOv3	0.929	0.10	0.893	0.15	0.929	0.10
YOLOv5	0.893	0.15	0.857	0.20	0.893	0.15
YOLOv7	0.964	0.05	0.929	0.10	0.929	0.10
Fast-RCNN	0.785	0.250	0.821	0.20	0.857	0.15

Table 6: The ablation study results

- The first stage of the anomaly query detection module can effectively differentiate between normal query and attack query traffic.
- The second stage, the active confirmation module, can effectively reduce the false positive rate.

Conclusion

- **Limitation**
  1. In OSD experiments, the defense effectiveness is only validated against a limited set of attack methods, which may still have limitations in terms of generalizability.
  2. OSD assumes that a single attacker can only launch attacks through a single account. However, if an attacker can simultaneously control multiple accounts for accessing services, OSD may not be able to detect all attacks.
  3. Due to the perturbation required by the active confirmation module on larger-sized targets, it still affects the user experience of legitimate users entering the active confirmation phase.
- **Conclusion**
  1. Through experiments on various task scenarios and model architectures, OSD has achieved efficient and accurate model theft detection.
  2. OSD is capable of adapting to the accuracy requirements of model extraction defense in cloud service scenarios while requiring low computational costs. It can satisfy the need for quick response to queries.